Machine Learning

Andrew Ng

RESOURCES

Syllabus
FAQ
Credits/Acknowledgments

Exercise 7: SVM Linear Classification

This exercise gives you practice with using SVMs for linear classification. You will use a free SVM software package called LIBSVM that interfaces to MATLAB/Octave. To begin, download the LIBSVM Matlab Interface (choose the package with the description "a simple MATLAB interface") and unzip the contents to any convenient location on your computer.

Then, download the data for this exercise: ex7Data.zip.

Installing LIBSVM

After you've downloaded the LIBSVM Matlab Interface, follow the instructions in the package's README file to build LIBSVM from its source code. Instructions are provided for both Matlab and Octave on Unix and Windows systems.

If you've built LIBSVM successfully, you should see 4 files with the suffix "mexglx" ("mexw32" on Windows). These are the binaries that you will run from MATLAB/Octave, and you need to make them visible to your working directory for this exercise. This can be done in any of the following 3 ways:

(1). Creating links to the binaries from your working directory

(2). Adding the location of the binaries to the Matlab/Octave path

(3). Copying the binaries to your working directory.

Linear classification

Recall from the video lectures that SVM classification solves the following optimization problem:

$$\min_{w,b} \quad \|w\|^2 + C\sum_i^m \xi_i$$

$$\begin{aligned} \text{subject to} \quad y^{(i)}(w^T x^{(i)} + b) &\geq 1 - \xi_i, \qquad i = 1, 2..., m \\ \xi_i &\geq 0, \qquad i = 1, 2..., m \end{aligned}$$

After solving, the SVM classifier predicts "1" if $w^T x + b \geq 0$ and "-1" otherwise. The decision boundary is given by the line $w^T x + b = 0$.

2-Dimensional classification problem

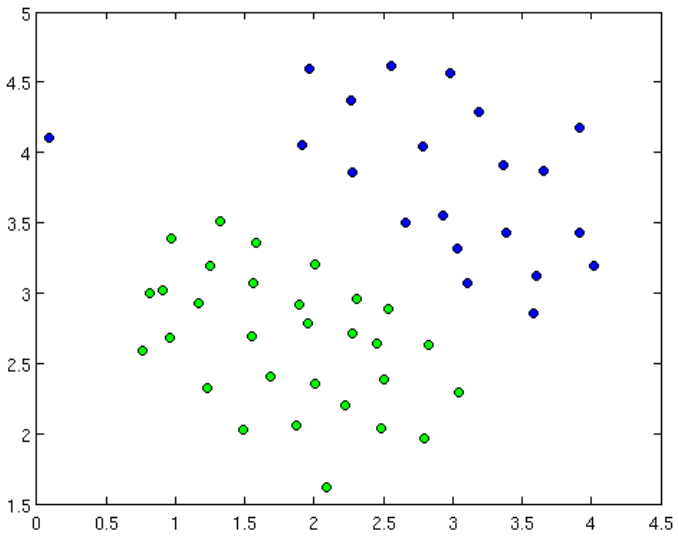
Let's first consider a classification problem with two features. Load the "twofeature.txt" data file into Matlab/Octave with the following command:

```
[trainlabels, trainfeatures] = libsvmread('twofeature.txt');
```

Note that this file is formatted for LIBSVM, so loading it with the usual Matlab/Octave commands would not work.

After loading, the "trainlabels" vector should contain the classification labels for your training data, and the "trainfeatures" matrix should contain 2 features per training example.

Now plot your data, using separate symbols for positives and negatives. Your plot should look similar to this:



In this plot, we see two classes of data with a somewhat obvious separation gap. However, the blue class has an outlier on the far left. We'll now look at how this outlier affects the SVM decision boundary.

Setting cost to C = 1

Recall from the lecture videos that the parameter C in the SVM optimization problem is a positive cost factor that penalizes misclassified training examples. A larger C discourages misclassification more than a smaller C.

First, we'll run the classifier with $C = 1$.

To train your model, call

```
model = svmtrain(trainlabels, trainfeatures, '-s 0 -t 0 -c 1');
```

The last string argument tells LIBSVM to train using the options

a. `-s 0`, SVM classification

b. `-t 0`, a linear kernel, because we want a linear decision boundary

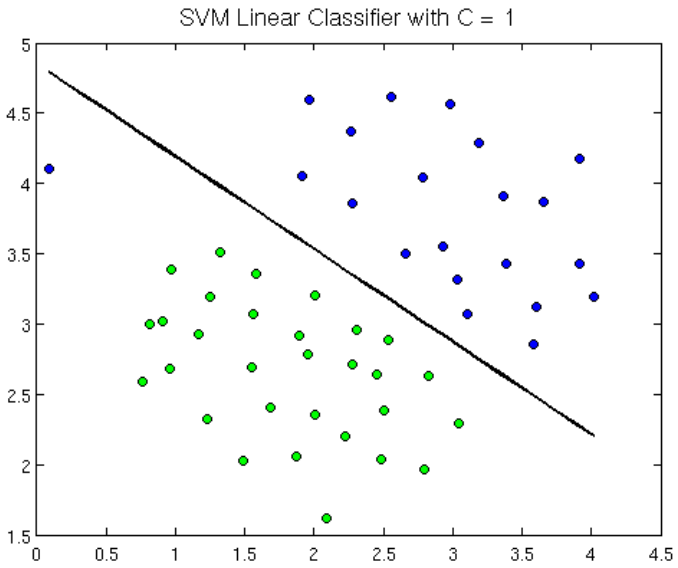
c. `-c 1`, a cost factor of 1

You can see all available options by typing "svmtrain" at the Matlab/Octave console.

After training is done, "model" will be a struct that contains the model parameters. We're now interested in getting the variables w and b. Unfortunately, these are not explicity represented in the model struct, but you can calculate them with the following commands:

```
w = model.SVs' * model.sv_coef;
b = -model.rho;
if (model.Label(1) == -1)
    w = -w; b = -b;
end
```

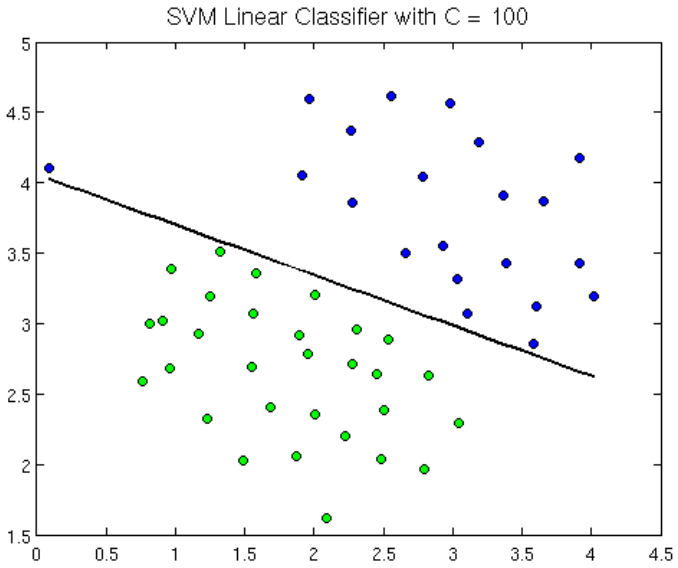
Once you have w and b, use them to plot the decision boundary. The outcome should look like the graph below.



With $C = 1$, we see that the outlier is misclassified, but the decision boundary seems like a reasonable fit.

Setting cost to C = 100

Now let's look at what happens when the cost factor is much higher. Train your model and plot the decision boundary again, this time with C set to 100. The outlier will now be classified correctly, but the decision boundary will not seem like a natural fit for the rest of the data:



This example shows that when cost penalty is large, the SVM algorithm will very hard to avoid misclassifications. The tradeoff is that the algorithm will give less weight to producing a large separation margin.

Text classification

Now let's return to our spam classification example from the previous exercise. In your data folder, there should be the same 4 training sets you saw in the Naive Bayes exercise, only now formatted for LIBSVM. They are named:

a. email_train-50.txt (based on 50 email documents)

b. email_train-100.txt (100 documents)

c. email_train-400.txt (400 documents)

d. email_train-all.txt (the complete 700 training documents)

You will train a linear SVM model on each of the four training sets with C left at the default SVM value. After training, test the performance of each model on set the named "email_test.txt." This is done with the "svmpredict" command, which you can find out more about by typing "svmpredict" at the MATLAB/Octave console.

During test time, the accuracy on the test set will be printed to the console. Record the classification accuracy for each training set and check your answers with the solutions. How do the errors compare to the Naive Bayes errors?

Hide Solution

Solutions

An m-file implementation of the two-feature exercise can be found here.

An m-file for the email classification exercise is here.

Classification accuracy

Here are the classification performance results that LIBSVM reports.

a. 50 documents: Accuracy = 75.3846% (196/260)

b. 100 documents: Accuracy = 88.4615% (230/260)

c. 400 documents: Accuracy = 98.0769% (255/260)

d. the complete 700 training documents: Accuracy = 98.4615% (256/260)

Here are the error comparisons with Naive Bayes:

	Naive Bayes	SVM
50 train docs	2.7%	24.6%
100 train docs	2.3%	11.5%
400 train docs	2.3%	1.9%
700 train docs	1.9%	1.5%

The conclusion from these results is that Naive Bayes performs better than SVM with less data, but SVM shows better asymptotic performance as the amount of training data increases.

© 2010-2012 Andrew Ng, Stanford University. All rights reserved.